Your name: ______________________ Your student number: _______

Department of Computer Science
CMPT 470-01/CMPT 816-01

Professor: Dr. K. A. Schneider

Midterm Examination
October 23, 2003

Time Limited: 60 Minutes

Total Marks: 60

This is a ***closed book*** *exam. Please write your answers legibly in the space provided* ***on this examination paper****. If you need more space, use the back of the page, but you should note that the space provided corresponds approximately to the length of answer desired. The answer booklets should only be used for rough work and will not be marked. Be sure to budget your time appropriately so you can* ***answer all questions****.*

Good Luck!

$\frac{51\frac{1}{2}}{60}$

6

13½

1 Short Answers [15 marks, 3 marks each]

1.1 The first law of software evolution was defined by M. Lehman as
"I - Continuing Change: An E-type program that is used must be continually adapted else it becomes progressively less satisfactory."

Please explain.

This means that if the system stays static while everything around it is changing then that system will be less satifactory than if it continues to adapt to new enviroments. It will seem less satisfactory because it will not be able to cope with the new requirements that the enviroment needs as well as the new functionality that the user requires.

3

1.2 Define 'Software Evolution' and discuss how it differs from 'Software Maintenance.'

Software Evolution is the processes involved with adapting software to meet the organization or business objectives in a cost effective way. Software maintenance is the modification of software after its been delivered to correct faults, improve performance or other attributes such as adapting to changed environments. Software evolution differs in that its on a bigger scale than software maintenance. Software maintenance is a subset of software evolution.

3

1.3 Software Architecture is used as a vehicle for stakeholder communication. What else is software architecture used for?

Software architecture is also used for capturing design decisions, its a way to manage change. It also provide an organizational structure, a way to break down the problem.

3

1.4 Compare and contrast 'conceptual software architecture' and 'concrete software architecture.'

Conceptual software architecture gives the overall view of the architecture but the important connections not necessarily every single connection. Concrete software architexture does show all of these connections.

2½

1.5 Compare and contrast reverse engineering and the software reflexion model.

Reverse engineering is the process of aquiring design abtractions from the implementation. Reflexion is also a way to extract design abstractions resulting in a reflexion model. Reverse engineering is a fact extraction process from which we can derive other facts

2

14½

2 Longer Answers [20 marks, 5 marks each]

2.1 D. Parnas has been quoted as saying "A sign that the Software Engineering profession has matured will be that we lose our preoccupation with the first release and focus on the long term health of our products." List five (5) techniques for ensuring the long term health of a software system.

1. Plan for change. → Information hiding → abstraction → object oriented.
2. Document everything.
3. Perform code reviews.
4. use principles of good design.
5. Plan for the retirement of the system.
   ↳ plan for what will happen.

2.2 List the refactoring steps to add a parameter to a method.

4½

- check the super classes and subclasses for the new method signiture thats about to be added. If there is one that's the same follow the following steps for each one.
- add the new method with parameter to the class and copy the old method's body to the new method and make changes to accomadate the new parameter.
- Compile
- call the new method from old method.
- Compile + test
- change all calls to the old method to the new method
- Compile + test
- delete or depreciate old method
- Compile + test.

2.3 Identify some of the difficulties in processing programming language source code when reverse engineering.

- Comments: In some languages comments can appear in very unusual places.
- Syntax: Some language syntax makes parsing difficult.
- Context specific scanning:
  ↳ This is where things can have different meaning depending on their location or use.
- incorrect code/input -> invalid
  ↳ the code given isn't correct.
  ↳ some compilers generate code for corrections.
- Naming resolutions: There can be ambiguities in which names refer to what -> UIDs can help this.
- Missing code - Code that isn't there is difficult to do anything with.

2.4 List five (5) code "smells" that you may look for when refactoring a program and briefly explain each of the five (5) code smells that you have listed.

- Lots of Comments: These are a sign that the code is too complex in that it must be explained.
- Long method: a method that is too long can lead to reduced ease of understanding of the code.
- Long class: a class that is too long.
- Lazy class: a class that doesn't appear to do much, most likely part of inheritance.
- Long parameter list in method.
  a method that has an unusually long parameter list is a source of complexity.

9

3 **Define each of the following terms. [10 marks, 2 marks each]**

3.1 Forward Engineering

This is the process of going from requirements to design to implementation. It can also use artifacts from an existing system for the requirements.

2 ✓

3.2 Reverse Engineering

This is the process of going from implementation to design to requirements. This is a fact extraction process. Includes design recovery.

2 ✓

3.3 Reengineering

This is the process of first reverse engineering and then forward engineering. Its useful for adding new requirements to an existing system.

2 ✓

3.4 Restructuring

This is the transformation of one representation to another at the same abstraction level. Usually involves modifying the control structure. not behavior

1

3.5 Design Recovery

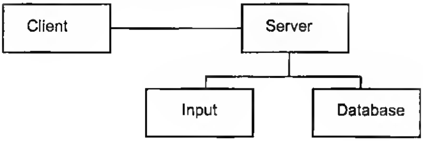
This is the extraction of design abstractions from:

- the system
- domain knowledge
- External knowledge
- Deduction.

2 ✓

3

4 **Software Architecture [6 marks, 2 marks each]**



4.1 What can we tell about the software system described in the above diagram?

- the client object knows the server.
- the server has two subcomponents Input, Database.

|

4.2 What can we NOT tell about the software system described in the above diagram?

- We can't tell how many clients can communicate with "how many" servers.
- we can't tell the direction of communication or the type of connections they have with each other

|

+ . . .

4.3 Does the above diagram show a software architecture? Explain.

- The diagram shows a type of software architectual.
- not all of the connections are apparent, communications are not clear. This is reminicent of a Live architecture, not everything is clear.

|

$6\frac{1}{2}$

5 Software Architecture Styles [9 marks, 3 marks each]

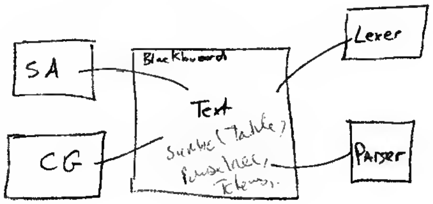
You have been asked to create the software architecture for a compiler. The compiler has four modules with the following functionality:

- a lexer (scans text into tokens and uses a symbol table to determine keywords)
- a parser (parses tokens into a parse tree and updates the symbol table)
- a semantic analyser (attributes a parse tree with semantic information and updates the symbol table)
- a code generator (generates code from the attributed parse tree and uses symbol table)

The software architecture of the compiler using a pipe and filter architectural style is as follows:

$$\begin{tikzcd}
{} \arrow[r, "\text{Text}"] & \boxed{\text{Lexer}} \arrow[r, "\substack{\text{Tokens}\\ \text{Symbol Table}}"] & \boxed{\text{Parser}} \arrow[r, "\substack{\text{Parse Tree}\\ \text{Symbol Table}}"] & \boxed{\text{SA}} \arrow[r, "\substack{\text{Attributed Parse Tree}\\ \text{Symbol Table}}"] & \boxed{\text{CG}} \arrow[r, "\text{Code}"] & {}
\end{tikzcd}$$

5.1 Draw the software architecture of the compiler using a blackboard architectural style.



2

5.2 Describe the criteria you would use to choose between the two software architectures.

- For DataFlow (pipe + Filter), I would use the fact that it seems like the data is going through a number of transitions. ← I'd pick this one!
- for Pipe + Filter I would use the fact that the processers seem to be sharing all of the data, The data seems important.

1½

5.3 Describe the advantages and disadvantages of the two software architectures.

DataFlow: (pipe+filter) adv: - easy to use
- easy to modify
- ability to perform statistical analysis.
- easy to extend + maintain
- ability to run concurrent processes.

dis: - Lead to batch processing → user interactively low.
- sensitive to input format.
- Streams may be similar
- leads to transferring of data by lowest common denominator.

3

Data-centered: (Blackboard) adv: - easy to add more clients
- separation of clients + data
- clients can be modified without effecting others

Dis: - performance issuses may result in clients being coupled.
- Creating the repository is complex, and difficult.